PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 :
G06F 11/30, 12/14, 15/173, H04L 9/00, 9/32

A1

(11) International Publication Number: WO 00/70464

(43) International Publication Date: 23 November 2000 (23.11.00)

(21) International Application Number: PCT/US00/12725

(22) International Filing Date: 9 May 2000 (09.05.00)

(30) Priority Data:

60/134,090	14 May 1999 (14.05.99)	US
60/144,319	16 July 1999 (16.07.99)	US
09/506,022	17 February 2000 (17.02.00)	US

(71) Applicant: L-3 COMMUNICATIONS CORPORATION [US/US]; 34th floor, 600 Third Avenue, New York, NY 10016 (US).

(72) Inventors: BARNETT, Bruce, G.; 64 Calhoun Drive, Troy, NY 12182 (US). HARTMAN, Michael, J.; 10 Spice Mile Blvd., Clifton Park, NY 12065 (US). BUSH, Stephen, F.; 9 Sable Terrace, Latham, NY 12110 (US). STAUDINGER, V., Paul; 2161 Morrow Ave., Niskayuna, NY 12309 (US).

(74) Agents: ROCCI, Steven, J. et al.; Woodcock Washburn Kurtz Mackiewicz & Norris LLP, 46th floor, One Liberty Place, Philadelphia, PA 19103 (US).

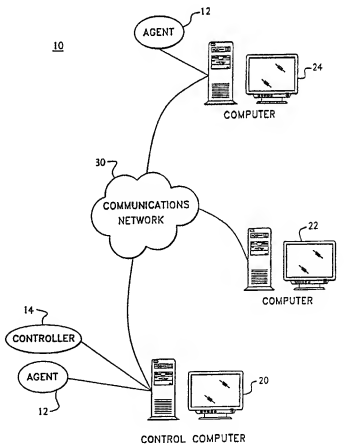
(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: OBJECT ORIENTED SECURITY ANALYSIS TOOL

(57) Abstract

Apparatus and methods for recognizing security threats on a data network (30) are disclosed. The invention can be implemented in hardware or software as an object oriented security analysis tool installed in each of the plurality of computers (20, 22, 24). The apparatus and methods of the present invention include designating one of the plurality of computers as a control computer (20), and installing and running an agent (12) on each of the plurality of computers (20, 22, 24). Each agent (12) monitors the computer on which it is installed to determine whether a security threat exists. If the agent (12) detects that a security threat exists, the agent (12) communicates to the controller (14) that a security threat exists.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/12725

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 11/30, 12/14, 15/173; H04L 9/00, 9/32
US CL : 713/200, 202; 709/223, 224

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 713/200, 202; 709/223, 224

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X ----- Y	ISS, Internet Scanner User Guide, Internet Security Systems, Inc., 1997, Version 5.2, pages 3,5,9,11,58-60,81,111	1,2,4-11 ---------- 3
Y	FARMER et al, The COPS Security Checker System, Purdue University Technical Report CSD-TR-993, September, 1991, pages 10-11	3

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 30 JUNE 2000	Date of mailing of the international search report 26 JUL 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer CHRISTOPHER REVAK James R. Matthews Telephone No. (703) 305-9618

Form PCT/ISA/210 (second sheet) (July 1998)*

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

BRS (USPAT, USOCR, DERWENT, JPO, EPO), IEEE, ACM, SCIENCE SERVER, PROQUEST DIRECT, WWW.ISS.NET, WWW.INSECURE.ORG, WWW.CERIAS.PURDUE.EDU

search terms: susceptible, illicit, malicious, attack, intrusion, breach, vulnerable, vulnerability, scan, examine, detect, inspect, inspection, security, scanner, analyze, analysis

OBJECT ORIENTED SECURITY ANALYSIS TOOL

Field of the Invention

This invention relates to computer data networks. More particularly, the invention relates to systems and methods for recognizing and reporting security threats on a computer data network.

Background of the Invention

Computer network security is a growing source of concern for businesses, especially for businesses that provide network services as a primary product. Current commercial security packages and services are costly and their effectiveness is not clearly understood. More precisely, security is implemented by policies, which often conflict with certain conveniences and functionality. Managers that set policies often do not understand the impact on functionality and potential threats.

Security tools are known in the art, but these tools are often unsuitable for certain applications. Typically, these security tools do not allow the integration of information from multiple sources. Some tools are not suitable for use with distributed networks. Others are hard to install. Still others cannot be upgraded remotely and, therefore, can be hard to maintain.

As more computers are added to a computer data network, security software that is capable of recognizing security threats on the network becomes more difficult to

implement. In general, network security is not well understood. A primary reason for this is that few security checking programs actually measure security, *i.e.*, recognize and grade security threats on the computers or systems being monitored. The tools merely provide a list of security threats, but fail to provide a way to determine whether security is getting better or worse.

Thus, there is a need in the art for the ability to easily and precisely understand the security vulnerabilities of a computer data network and the manner in which security safeguards strengthen those weaknesses.

Summary of the Invention

The present invention satisfies these needs in the art by providing systems and methods for recognizing security threats on a data network comprising a plurality of computers. Preferably, the present invention is implemented in hardware and/or software as an object oriented security analysis tool installed in each of the plurality of computers. The systems and methods of the present invention comprise: designating one of the plurality of computers as a control computer; installing and running a controller on the control computer; and installing and running an agent on each of the plurality of computers. Each agent continuously monitors data on the computer on which it is installed to determine whether a security threat exists on the network. If the agent determines that a security threat exists, the agent communicates to the controller that a security threat exists, and the controller notifies a network administrator.

Brief Description of the Drawings

The foregoing summary, as well as the following detailed description of the preferred embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings an embodiment that is presently preferred, it being understood, however, that the invention is not limited to the specific methods and instrumentalities disclosed.

Figure 1 depicts a system according to the present invention for recognizing security threats on a data network.

Figure 2 provides an object model of a preferred embodiment of a system

according to the present invention.

Detailed Description of Preferred Embodiments

Object oriented programming systems and processes, also referred to as "object oriented computing environments," have been the subject of much investigation and interest in state of the art data processing environments. Object oriented programming has experienced growing popularity with application developers primarily due to the ability of this type of programming to reuse code.

As is well known to those having skill in the art, object oriented computing environments are made up of a large number of "objects." An object in an object oriented environment consists of related pieces of code and data. More specifically, an object comprises a data structure, also referred to as a "frame," and a set of operations or functions, also referred to as "methods," that can access the data structure. The frame has a plurality of "slots," each of which contains an "attribute" of the data in the slot. The attribute can be a primitive (*e.g.*, an integer or string), or an object reference (*i.e.*, a pointer to another object). The object generally has a library of methods that are unique to the object and give the object its specific characteristics.

The "class" of an object defines a grouping based on one or more properties, and an "instance" is a single item in a class. For purposes of illustration, a class could be considered as analogous to professions, *e.g.*, engineers, lawyers, doctors, *etc.*, with an instance being analogous to a particular individual in a class. An object will usually be manifested in a plurality of instances. Each instance contains the particular data structure for a particular example of the object. A class defines methods for constructing new instances. "Instance variables" define the particular features of an instance, such as an individual's salary might be a feature of one of the instances in the example discussed above. These instance variables may be defined or may be empty, *i.e.*, awaiting definition. An instance's data structure is described by its collection of instance variables. Pointers are used to point to a structure in object form.

In an object oriented computing environment, data is processed by requesting an object to perform one of its methods by sending the object a "message." The receiving object responds to the message by choosing the method that implements the message name, executing this method on the named instance, and returning control to the

calling high level routine along with the results of the method. A class defines instance methods which define how an instance behaves and reacts to incoming messages.

In some computer systems, it is important to allow import and export of data between the object oriented computing environment and an external computing environment. The external computing environment can be a conventional, functionally programmed, computing environment, or it may be another object oriented computing environment. Typical interfaces between an object oriented environment and an external environment may include an interactive host user interface, a batch interface, a remote work station user interface, or other well known interfaces between computing environments.

In importing and exporting data between an object oriented computing environment and an external computing environment, an interface is typically defined and generated between the object oriented computing environment and the external computing environment. The interface typically includes mechanisms for validating the imported and exported data, and for converting the imported and exported data to a form that is usable by the importing or exporting system.

Object Oriented Security Analysis Tool

Figure 1 depicts a system 10 according to the present invention for recognizing security threats on a data network. A distributed computer data network comprises a plurality of computers 20, 22, 24 interconnected to one another via a communications network 30. Communications network 30 can be, for example, a local area network (LAN), a wide area network (WAN), an intranet, or the Internet. Computers 20, 22, 24 are capable of communicating with one another using agreed-upon protocols. In a preferred embodiment of the present invention, an object oriented security tool is installed on the network for the purpose of monitoring the computers on the network to recognize whether a security violation has occurred or could occur, *i.e.*, whether a security threat exists on the network.

An agent 12 is installed on each monitored computer 20, 24 and is continuously active thereon. A "centralized system," or controller 14, is installed on a control computer 20 (which is selected from among the computers on the network). The controller 14 communicates with each of the agents 12, and, in a preferred embodiment,

transfers software updates to the agents 12 to keep them current. In this way, the controller 14 prevents the agents 12 from becoming obsolete.

The controller 14 also maintains a database of information about the monitored computers 20, 24 based on an object model. Figure 2 provides an object model of a preferred embodiment of a system according to the present invention. The object model includes classes of objects and the relationships between objects. It should be understood that the object model shown in Figure 2 is but one embodiment of the present invention, and that variations and modifications can be made thereto without departing from the spirit and scope of the present invention.

An object class describes one or more objects which correspond to hardware and software related information. For example, the class of "Host" describes certain characteristics of the monitored computers, or hosts, as well as routines used to query and change those characteristics. In the Host class, there are one or more sets of information, where each set of information corresponds to one object, or instance of that object. In this case, there is a set of information that describes one monitored computer on the network with a particular host name.

In a preferred embodiment, an object oriented security tool according to the present invention comprises the following plurality of object classes. Not all classes are required for every algorithm.

Object class "Host" 40 includes instances of monitored computers on the network. Object class "Account" 42 includes a description of a user on a monitored computer. Object class "File" 44 includes a computer file on a monitored computer. Object class "Directory" 46 includes a file directory on a monitored computer.

Object class "OS" 48 includes a particular operating system from a vendor, including revisions number. Object class "Patch" 50 includes a package from the vendor to upgrade the operating system. Object class "Signature" 52 includes a unique representation for a file, to verify that the file is the one expected. It uses cryptographic techniques to identify each file, *i.e.*, a one-way hash of the contents of the file.

Object class "Vulnerability" 54 includes a description of a mechanism where the privileges of one account can access another. Object class "Service" 56 includes a description of a particular function on a monitored computer that can be controlled by

modifying the configuration of the monitored computer. Object class "Policy" 58 includes a description of a policy that protects monitored computers from threats, and is enforced by enabling, disabling, or modifying particular services on a monitored computer.

Object class "Function" 60 includes a description of the function that a service has that is used to find alternate services that can provide the same function. Object class

"Threat" 62 includes a particular attack mechanism that can be used to break into a monitored computer. By enforcing particular policies, threats can be neutralized.

In a preferred embodiment, a system according to the present invention includes a mechanism to "browse" the data by traversing the relationships between individual objects. That is, given one object (*e.g.*, a host), the system provides a way to find all related objects of another class. If the class is Accounts, for example, the system can find all accounts "belonging" to a particular host. The traversal from one object to another can be two-directional. Assume that one host has a relationship to many accounts, *i.e.*, there is a one-to-many relationship between object class Host and object class Account. Given an account, it is possible to find the host on which the account resides (in this case, for example, each account resides on only one host). Conversely, it is possible to find all of the accounts that reside on a host (there are many accounts on the host). This relationship may or may not be named. The host-to-many-accounts relationship is the only relationship between these two particular objects; therefore, there is no need to name it. The vulnerability object, however, has two relationships between the object Vulnerability and the object Account. A mechanism is needed, therefore, to distinguish between these two relationships. In this case, the two relationships are called Attacker and Victim.

There are two currently preferred embodiments of the present invention. In one embodiment a fixed number of object classes is used, such as the list described above. In an alternate embodiment, a meta-model is used so that any object class, as well as any set of relationships, can be specified. This alternate embodiment allows the classes to be defined and modified at run time.

The agents continuously monitor, and gather information from, the plurality of monitored computers, and consolidate all of the information into a centralized database maintained by the controller. Multiple databases can be organized in a hierarchal fashion,

allowing for systems to be responsible for sub-groups.

The database contains information about each individual instance of an object (*e.g.*, each account on each system), and also includes information that links individual objects to other objects. For example, each vulnerability object can be associated with three other objects. The first indicates the account that is attacked. The second indicates the account that can

attack the first account. The third is a file that is used for this attack, called a vector. In this example, each vulnerability object has two references to account objects, and one to a file object.

Standard algorithms to gather information are known; however, the output is typically a list form, and the data is not integrated. The system of the present invention integrates the information by using instances of objects. Once the database is constructed (which can be done by using several algorithms, in any order), and information is gathered and merged, other algorithms can be used to browse the database.

The following notation, "class1 ---> class2 ---> class3," will be used in the discussion below to indicate that information about a particular instance of an object of type "class1" is used to get a set of references to related objects of type "class2". These in turn are used to retrieve a set of objects of type "class3".

For performance reasons, a preferred embodiment of the present invention can use cached data when retrieving information from remote sites. Conversely, if information is needed, and missing, the system can retrieve it, and the algorithm does not need to know this is happening.

The following discussion includes the descriptions of several algorithms that traverse the objects in the database by using the relationships between objects. Each algorithm provides a unique function, and the inventors do not know of any other system which can implement any of these functions. In a preferred embodiment of the present invention, however, only one such function is implemented. Many of the potential algorithms are discussed for the sake of completeness, although the algorithms that can be used with the present invention are not limited to those discussed. Using object model based technology allows more algorithms to be easily implemented. Therefore it is important to protect the core system.

Modification Detection

The software can be used to monitor the integrity of each file on the system, so tampering can be detected. This is done as follows. The host type (Host) is retrieved, and the operating system type (OS) is learned. From this a list of suitable patches (Patches) is retrieved, and from this, a list of files (Files) is obtained. Each file has a signature that gives the hash of the file. The agent then calculates the hash of the file, and returns information indicating whether the hash is correct. If so, the correct version is present. Otherwise, the file is incorrect, and the system can determine that a patch as been applied. The relationships traversed during this algorithm are

Host ---> OS ---> Patches ---> Files ---> Signatures.

Remote Patch Maintenance

This is similar to the previous example, in that the following chain of objects can be traversed: Host ---> OS ---> Patches ---> Files ---> Signatures. From this, a system can be tested to see if all of the patches have been applied. If not, the patch can be identified. In a preferred embodiment, the patch is optionally transferred to the remote system and installed.

Vulnerability Chains

By following a chain of vulnerabilities, it is possible to determine which accounts are vulnerable to attack. The data traversal might be: Account1 ---> Vulnerability1 ---> Account2 ---> Vulnerability2 ---> Account3, although the chain can be much longer.

Once the chains have been identified, and examined, the system can report all of the accounts that a particular account can access. It can be used, therefore, to determine whether a particular account is a security risk. By working backward, it is easy to learn which accounts can access a particular account. This can be used in threat analysis. Another algorithm can analyze a large number of chains, and determine the vulnerabilities that permit the largest number of attacks. A fourth can identify the fewest number of cuts necessary to break a chain into two pieces, protecting a particular account. This can be considered a way to verify that an "air gap" exists between sections. An "air gap" is a separation between two systems. In this case, an air gap between two machines indicates that if one machine has been compromised, the other one is not necessarily compromised. If a vulnerability connected two machines, no air gap is present, and it is possible for an

intruder to gain access to the other machine through the vulnerability.

Policy Verification

By traversing the data in this direction: File ---> Service ---> Policy ---> Threat, it is possible to determine the threats to which particular systems are vulnerable. This can be used in an attack scenario, where potential weaknesses are analyzed, and coordinated attacks are planned.

Policy Evaluation

This is a variation of the above algorithm wherein the current policy can be learned to enable a manager to modify the policy to protect against threats. This will identify services that are in conflict with the policy. This algorithm can be used, therefore, to determine the effects of changing the security policy.

Policy Enforcement

By traversing the data in this direction: Policy ---> Service ---> File, it is possible to determine which policies are being followed, and which are not. If files are configured the wrong way, they can be modified to enforce the policy. This allows a site to set a policy, and then automatically to enforce the policy. It also allows someone who sets policy to see what functionality is lost by that policy. Therefore, this algorithm allows someone who is non-technical to understand the trade-offs between security, functionality, and convenience.

Genetic Variation

It is useful for a large site to be resilient to attacks. One way to do this is to permit variation among the different systems in a network. Some services may be optional, or have equivalent services. This is determined by understanding the relationship between services and functions. A function might be "remote access," for example, and multiple services can provide this function. A system can make sure that variations exist across a series of machines using this information. A policy manager can permit this variation, and make sure it is enforced. If an attack does occur, this algorithm will make sure that not every system is vulnerable in the same manner, because not every system runs exactly the same set of services. Redundant systems can be managed using this algorithm.

Intrusion Detection

An intrusion detection system can report which systems seem to be infected, by noticing unusual behavior. By querying the database, a system can identify which services a set of machines have in common, and therefore identify the vector used to spread a viral attack on a network of computers. The relationship traversal would be Host ---> Service --> Policy ---> Threat.

If all infected systems (*i.e.*, those systems identified because of anomalous action) have the same services, the vector can be identified as that service. If a set of services are involved, then the system can look for a common threat to which all services are vulnerable. Therefore the mechanism used to spread the virus can be identified, or eliminated by negative evidence.

Intrusion Detection Reaction

Once the vector has been identified by an intrusion detection system, the system can identify other systems that are vulnerable to the same attack, by searching for systems with the same services and the same architecture. A security manager can then disable the vulnerable services, and only the vulnerable services, to keep the systems up and running while under attack. The traversal could be Threat ---> Policy ---> Service ---> Host.

Currently, the network administrator of a system under attack either ignores the attack, or shuts down the entire system. If the attack mechanism is known to be the mail service, this system can ask each system running the mail service to disable it, allowing the systems to remain functional, without allowing the attack to spread. A similar algorithm can be used to disable only those systems with a particular version of a service. For example, a particular version of the mail service might have a vulnerability. All systems with this version can be identified, and then the services can be disabled. The object traversal might be: Signature ---> OS ---> Host ---> Service ---> File. Once the file is identified, it can be disabled, removed, fixed, *etc.*

Dynamic Repair

A system under attack can be asked to disable all services, using the following traversal: Host ---> Service, and asking each service to shut down. The system can then kill all of the running processes except for the critical subset needed to complete the task. Then the system can examine each file and directory for improper modification. If the file is the wrong version, it can be replaced. Then the system can enable some or all

of the services, by reversing the first procedure. This will bring a system under attack back to being operational.

Those skilled in the art will appreciate that numerous changes and modifications may be made to the preferred embodiments of the invention and that such changes and modifications may be made without departing from the spirit of the invention. It is therefore intended that the appended claims cover all such equivalent variations as fall within the true spirit and scope of the invention.

WE CLAIM:

1. A method for recognizing security threats on a network comprising a plurality of computers, comprising:

designating a computer from the plurality of computers as a control computer;

executing a controller on the control computer; and

executing an agent on each of the plurality of computers;

wherein each agent monitors the computer on which it is installed to determine whether a security threat exists on the network, and if a security threat exists, communicates to the controller that the security threat exists.

2. The method of claim 1, wherein the controller notifies a network administrator that the security threat exists.

3. The method of claim 1, wherein the agent determines whether the security threat exists on the network by determining whether a file has been modified on the computer on which the agent is installed.

4. The method of claim 1, wherein the agent determines whether the security threat exists on the network by determining whether a patch has been applied to a file on the computer on which the agent is installed.

5. The method of claim 1, wherein the agent determines whether the security threat exists on the network by determining whether an account on the computer is vulnerable to attack.

6. The method of claim 1, wherein each agent monitors the computer on which it is installed to determine threats to which the computer is vulnerable.

7. The method of claim 1, wherein each agent monitors the computer on which

it is installed to determine a current security policy.

8. The method of claim 7, wherein each agent monitors the computer on which it is installed to determine effects of changing the current security policy.

9. The method of claim 7, wherein each agent monitors the computer on which it is installed to determine whether the current security policy is being enforced.

10. The method of claim 1, wherein each agent monitors the computer on which it is installed to identify a vector that can be used to spread a viral attack on the computers.

11. Apparatus for recognizing security threats on a network comprising a plurality of computers, comprising:

a controller; and

a plurality of agents in communication with the controller,

wherein each agent is installed on a respective computer of the plurality of computers, and monitors the computer on which it is installed to determine whether a security threat exists on the network, and if the agent detects that the security threat exists, the agent communicates to the controller that the security threat exists.

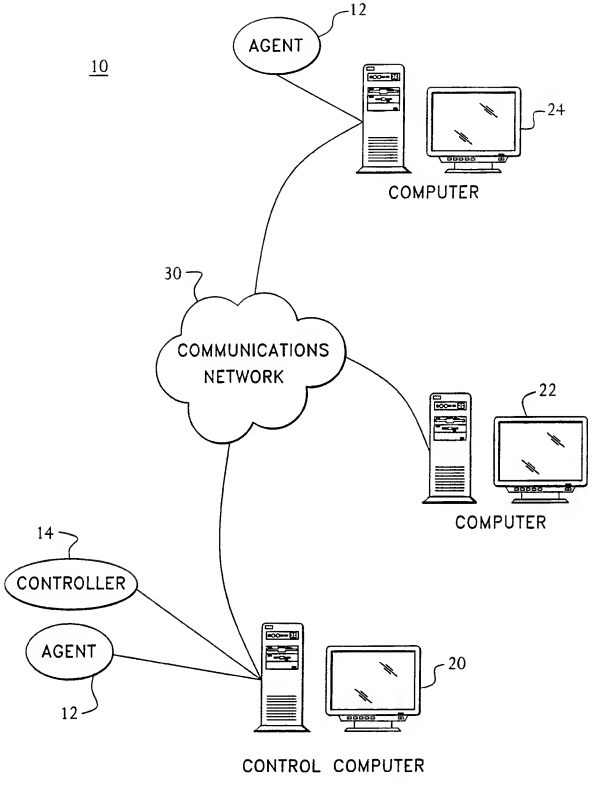

10
AGENT
12
24
COMPUTER
30
COMMUNICATIONS NETWORK
22
COMPUTER
14
CONTROLLER
AGENT
12
20
CONTROL COMPUTER

FIG. 1

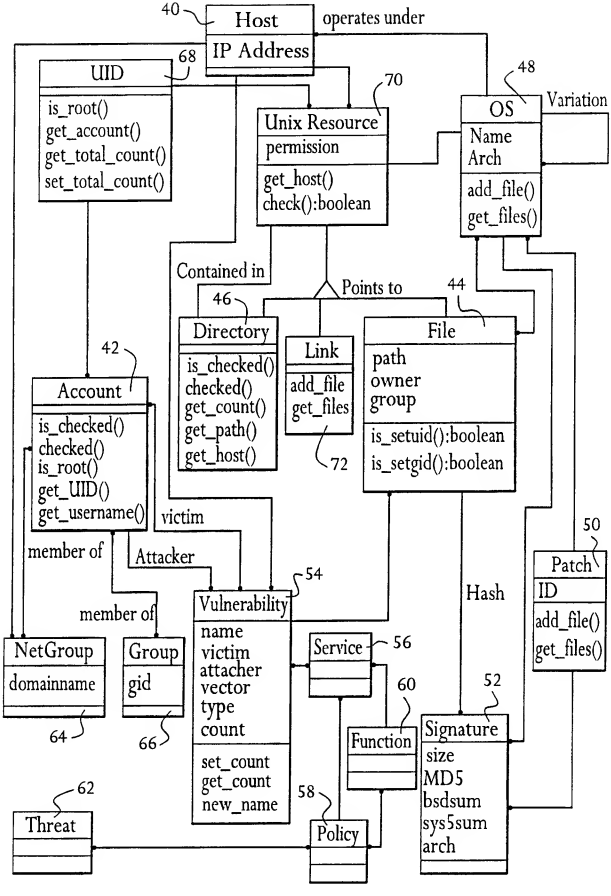

40
Host
IP Address
operates under
68
UID
is_root()
get_account()
get_total_count()
set_total_count()
70
Unix Resource
permission
get_host()
check():boolean
48
OS
Name
Arch
add_file()
get_files()
Variation
Contained in
Points to
46
Directory
is_checked()
checked()
get_count()
get_path()
get_host()
Link
add_file
get_files
44
File
path
owner
group
is_setuid():boolean
is_setgid():boolean
72
42
Account
is_checked()
checked()
is_root()
get_UID()
get_username()
victim
member of
Attacker
50
Patch
ID
add_file()
get_files()
54
Vulnerability
name
victim
attacher
vector
type
count
set_count
get_count
new_name
Hash
member of
NetGroup
domainname
64
Group
gid
66
Service
56
60
Function
52
Signature
size
MD5
bsdsum
sys5sum
arch
62
Threat
58
Policy

FIG. 2